# 《Java 特性篇》

## java语言有什么特性，继承有什么用处，多态有什么用处

答：java有八大特性：

1.1 面向对象语言，具备抽象、封装、继承、多态特性。

1.2 平台无关性，jvm屏蔽了操作系统差异，使得java语言通过字节码在不同操作系统的jvm上被差异化解释，实现java语言的平台无关性。

1.3 解释型语言，不同于C++、C语言，编译生成针对CPU的机器码，java语言编译生成字节码，会按句翻译执行。在jit即使编译技术加持下，可以将字节码转化成高效的本地机器码，提高执行效率。

1.4 多线程，java内置多线程支持，比C++语言更加方便，不需要调用操作系统多线程功能接口即可完成程序设计。

1.5 安全，java小程序运行在java环境中，不允许它访问计算机的其他部分。

1.6 动态编译，由于java类来自于开发者编写、类库引入，类是动态加载的，可以在分布式环境中动态地维护程序和类库，不用像C++，每次类库升级需要重新编译整个程序。

继承的作用：很方便地复用代码+不破坏现有代码的结构和功能。多态的用处：分离做什么和怎么做，进一步分离接口和实现。不仅可以改善代码组织结构和可读性，而且可扩展性也很强。

## 反射是什么，在哪里用到，怎么利用反射创建一个对象

在运行期间打开和检查java类，并提供获取方法、构造器和字段的接口的类库，在跨网络远程调用、基于RAD可视化编程中用到；

根据class名称获取类字节码对象，通过newInstance()调用无参构造器；getConstructor获取有参构造器，并完成对象构造。

2.1 Class<?> cls = Class.forName("class.name"); Object obj = cls.newInstance();

2.2 Class<?> cls = Class.forName("class.name"); Construct<?> con = cls.getConstructor(String.class,String.class); Object obj = con.newInstance(str1,str2);

## 对象加载的过程，属性先加载还是方法先加载

父类静态属性 (可以是对象) 和静态代码块，看其在类中的先后顺序

子类静态属性和静态代码块 ，看其在类中的先后顺序

父类非静态属性和非静态代码块 ，看其在类中的先后顺序

父类构造方法

子类非静态属性和非静态代码块 ，看其在类中的先后顺序

子类构造方法

规律：

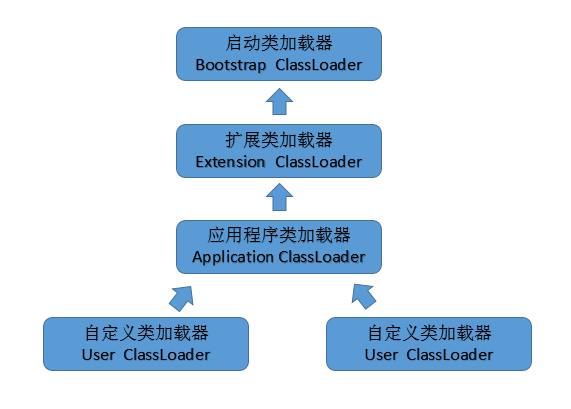
1.静态属性和代码块，跨类高优先级；

2.非静态属性和代码块同类中优先级高于构造函数；

3.静态和非静态的属性和代码块的加载顺序，同对应的定义顺序。

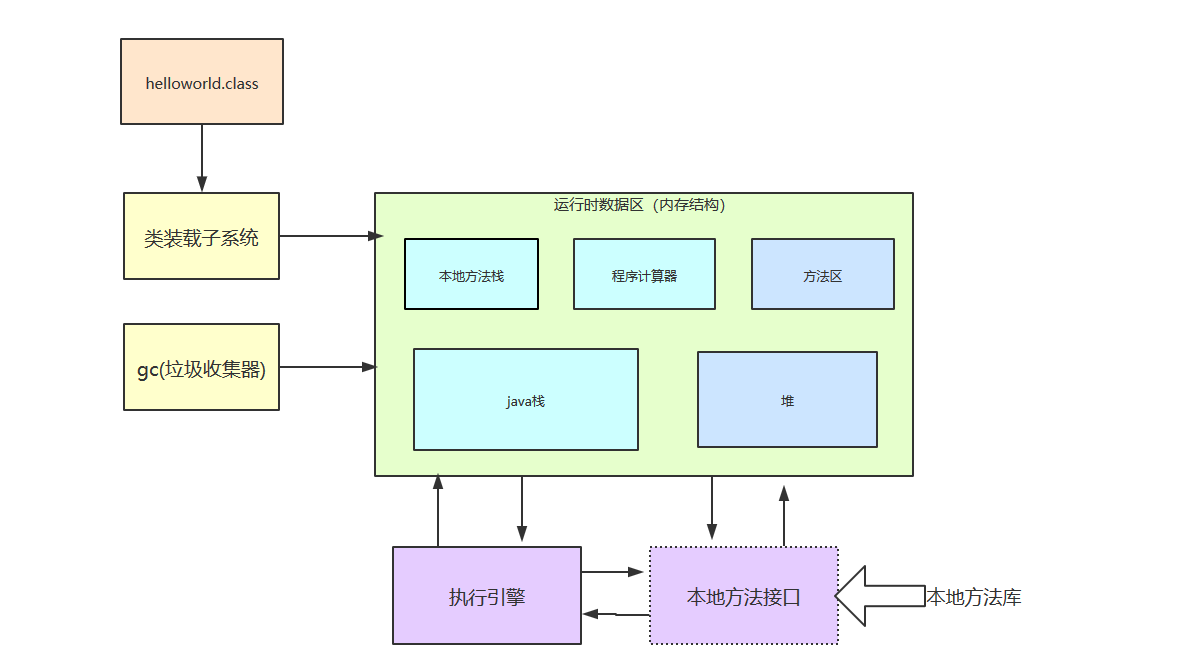
## 垃圾回收机制与jvm结构

**类加载：**java虚拟机是由启动类加载器加载的，负责执行java程序；java类加载器的结构：启动类加载器-->扩展类加载器-->应用程序类加载器-->用户自动定义加载器，满足双亲委派原则，保护java核心库类不被轻易篡改、避免类重复加载。



java类实例相同的判断：1.相同的类路径 2.同一个类加载器(避免不同虚拟机加载同一个class文件的情况)

**内存结构：**



**本地方法栈**

1. 线程私有
2. 存储的是本地方法的栈帧

**程序计数器**

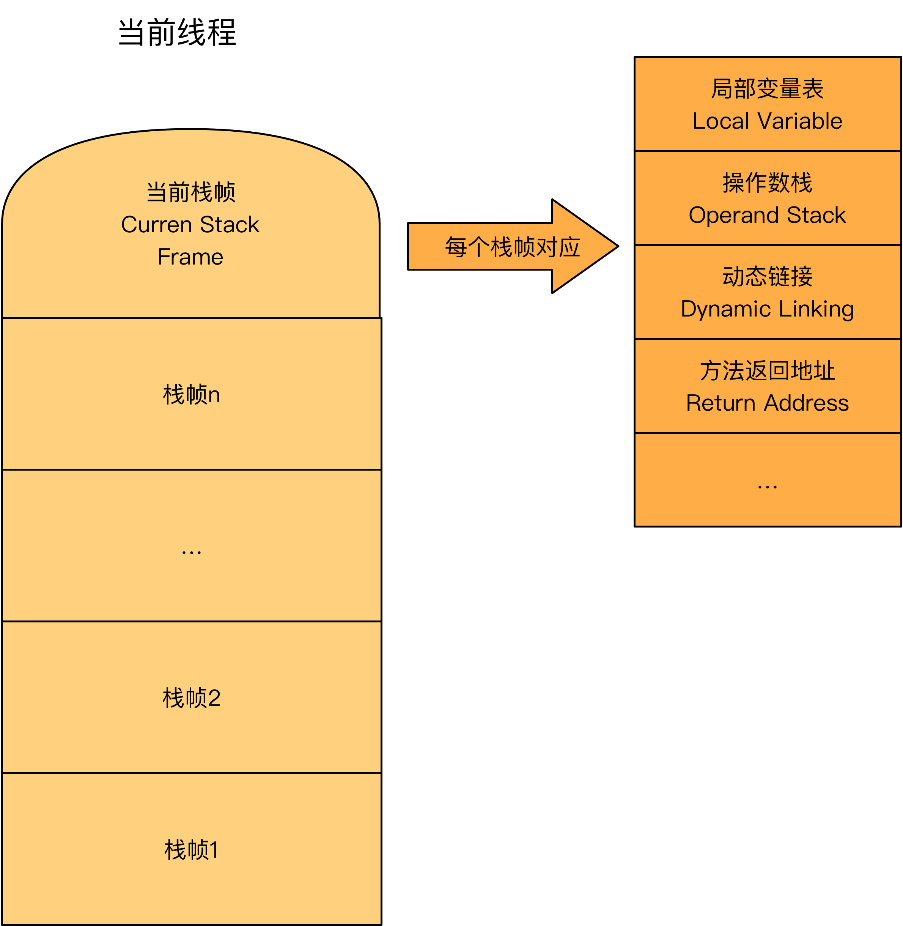
1. 线程私有
2. 当前线程所执行的字节码的指示器，
3. Java方法：存储的字节码指令地址；native方法，计数器值为空。
4. 唯一没有oom异常的数据区，因为只需要存储下一条指令地址，不会存在空间扩张。

**方法区**

1. 线程共享
2. 存储加载的类信息、常量池、静态变量

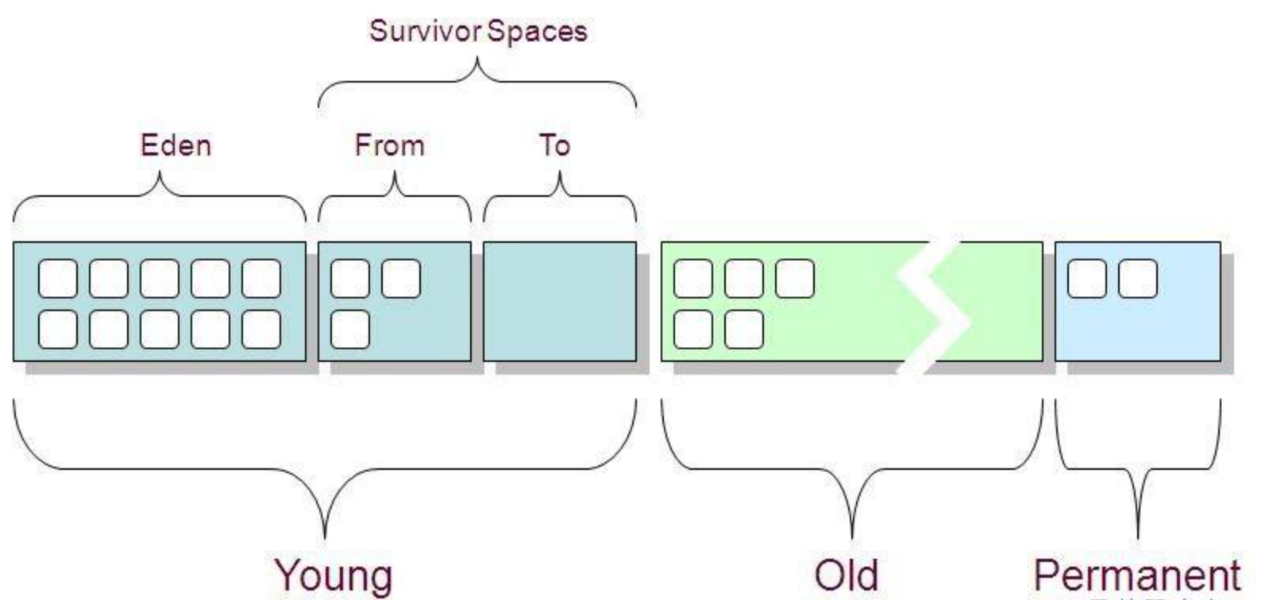
**线程栈**

1. 线程私有
2. 保存方法执行的栈帧
3. 栈帧存储了方法的局部变量表、操作数栈、动态链接和方法返回地址

****

**堆**

1. 线程共享
2. 存储对象实例
3. 物理空间可以不连续
4. 堆中的代划分

****

Eden满了，会进入survivor区进行迭代(复制算法)，达到迭代的阈值没销毁的对象，会进入老年区，老年区无法放入，进行MajorGC (标记—整理算法)。永久代放置静态变量和常量、类信息等。

# 《设计模式》

## 1.代理模式与装饰模式的区别，手写一个静态代理，一个动态代理

A.代理模式，主要用于为跨进程访问、安全控制、开销很大的对象提供访问层。++

B.装饰模式，用于在不破坏现有类结构的前提下，对类对象就行功能追加，相对继承更加灵活、方便。

C.静态代理, 代理类和目标类相对简单单一，代理类对目标类的访问控制、执行控制等进行增强。

D.动态代理，java运行期间，根据代码业务需要，通过JDK或CGLIB提供的接口(反射机制)，为指定的目标对象，生成对应的代理对象。

代码见Examples-->DynamicProxy

# 《Android知识》

1. 自定义View,事件分发机制讲一讲、

事件分发机制：activity🡪viewGroup—>View

* 1. interapt只有groupView存在
  2. keydown事件按照树深度优先(从右向左遍历)
  3. keyup、move、cancel事件都是按照事件响应链进行执行

1.4 自定义view事件响应顺序：onTouchListener > onTouchEvent > onLongClickListener > onClickListener

2. http与https有什么区别

2.1 超文本传输协议，用于web客户端和服务端通信

2.2 安全套接字层超文本传输协议，具备传输加密、服务器身份验证的功能。是http的安全版，由于传输过程中握手复杂，会导致页面加载时长比http长近50%，同时功耗也会更高。

* 1. Ssl证书一般需要付费。

1. Activity启动模式，以及各启动模式生命周期问题
2. standard：无限新增
3. singleTop：栈顶不新增；否则新增
4. newtask：清空activity上面所有其他activities，或者新开一个task
5. newinstance：全局唯一，走onNewIntent
6. onstart和onresume区别是是否在前台，都是可见的。
7. 静态方法，静态成员为什么不能继承

--静态方法和成员，属于类，存储在方法区，在编译期间就已经确定了存储位置和内容，通过父类的指针访问的方法和成员就是父类的相关资源，子类覆写会重新开辟空间，不存在继承的说法。

1. Activity怎么启动Service，Activity与Service交互，Service与Thread的区别
   1. startService或者bindService
   2. 动态广播监听、系统属性或SP、Messenger 或aidl。
   3. Service是安卓四大组件之一，一般负责执行一些比较耗时、耗性能的独立业务；Thread是java世界的最小执行单元，一般也会执行一些独立的业务，线程的生命周期独立于启用它的activity或service；service业务执行一般是在主线程中，intentService例外，它通过handlerThread新建了线程执行onHandleIntent中的方法，进行更方便的控制，包括多次启动退出。

6.介绍一下android动画

6.1 按照技术实现分类：补间动画(动画插值器控制速度)，可绘制图像动画(矢量图像、图像序列)、属性动画(属性包括属性值等)。

补间动画和属性动画的差别：补间动画仅作用于view，属性动画可以作用于view和object。

6.2 Lottie开源动画库，解析jason完成动画绘制。基于Drawable绘制接口完成。

7.Launcher启动App的流程，中间有几种跨进程通信(socket)

7.1 launcher调用AMS startActivity，启动app页面

7.2 AMS检查进程是否存在，存在则直接执行Activity启动；否则向zygote申请fork新进程。

7.3 应用进程启动后，执行ActivityThread中的main方法。Main函数执行attachApplication，向AMS绑定进程和应用。

7.4 AMS收到attachApplication后，调用IApplicationThread.aidl接口中的bindApplication和scheduleTransaction,分别通知应用创建Application和启动第一个activity界面。

7.5 scheduleTransaction会启动第一个activity，创建activity、phoneWindow、DecorView对象，执行activity的启动阶段流程(create、start、resume)函数。

7.6 有三种跨进程调用：aidl、binder、socket

参考：https://blog.csdn.net/followYouself/article/details/125176288

8.Handler通信，Binder通信

9.你碰到过什么内存泄漏，怎么处理

10.RXJava怎么切换线程

11.Fragment hide show生命周期

12.平常有用到什么锁，synchronized底层原理是什么